*Journées de l'Association Française d'Informatique Graphique, Reims Image 2014*

# Modélisation de terrains par primitives

Jean-David Génevaux [1], François Grosbellet [1,2], Éric Galin [1], Adrien Peytavie [1], Éric Guérin [1], Cyril Briquet [1], Bedřich Beneš [3]

[1]Université de Lyon, LIRIS, CNRS, UMR5205, France [2]Université de Limoges, XLIM, CNRS, UMR7252, France [3]Purdue University, USA

**Résumé**

*Nous proposons un modèle de terrain hiérarchique et compact permettant de représenter des scènes complexes. Ce modèle de représentation s'inspire des surfaces implicites à squelettes et définit une fonction d'élévation sous la forme d'un arbre de construction. Les feuilles sont des primitives décrivant des morceaux de terrains à différentes échelles (montagnes, fleuves, ...) et les nœuds internes sont des opérateurs de combinaison. L'élévation d'un point est calculée en traversant la structure d'arbre et en combinant les contributions de chaque primitive. La définition des feuilles et des opérateurs garantit que la fonction d'élévation résultante est Lipschitzienne, ce qui permet d'accélérer les calculs de visualisation en utilisant un algorithme de* sphere tracing.

***Mots Clés :*** *modélisation de terrains, phénomènes naturels, modélisation procédurale, surface implicite*

*We propose a compact hierarchical procedural model that combines feature-based primitives to create complex continuous terrains. Our model is inspired by skeletal implicit surfaces and defines the terrain elevation by using a construction tree whose leaves are primitives describing terrain fragments, and whose inner nodes include operations that combine its sub-trees. The elevation of a point is evaluated by traversing the tree and by combining the contributions of each primitive. The definition of both leaves and operators guarantees that the resulting elevation function is Lipschitz which enables us to speed up sphere tracing and surface adaptive tesselation algorithms.*

***Keywords:*** *terrain modelling, natural phenomena, procedural modelling, implicit surface*

## 1. Introduction

La modélisation de terrains a été un sujet de recherche actif en informatique graphique depuis de nombreuses années. Malgré les progrès considérables dans ce domaine, il reste de nombreux problèmes de contrôle et de performance.

Les techniques actuelles peuvent être classées en différentes catégories : la génération procédurale, les simulations, ou les algorithmes de construction à partir d'exemples ou d'esquisses. Les méthodes procédurales sont efficaces sur le plan calculatoire mais ne fournissent pas de contrôle précis sur la structure du terrain obtenue. Les méthodes de simulation d'érosion fournissent des résultats géologiquement corrects, mais ne permettent pas un bon contrôle et nécessitent des temps de calcul importants. Les méthodes d'édition à partir d'esquisses nécessitent un contrôle manuel pouvant être fastidieux. Enfin les méthodes de synthèse à partir d'exemples ne permettent de reproduire qu'un seul type de relief homogène aux images données. Il n'existe pas, à ce jour, d'algorithme permettant une modification interactive de terrains complexes avec un haut niveau de détails au sol, et permettant un placement rapide et intuitif d'éléments caractéristiques géologiques et géographiques en utilisant des opérations d'édition simples. Un problème connexe vient du fait que beaucoup de ces algorithmes ne permettent pas de représenter les terrains à différentes échelles : les terrains sont modélisés sur des structures de grilles qui représentent des reliefs à une précision fixe. Ces terrains sont souvent sto-

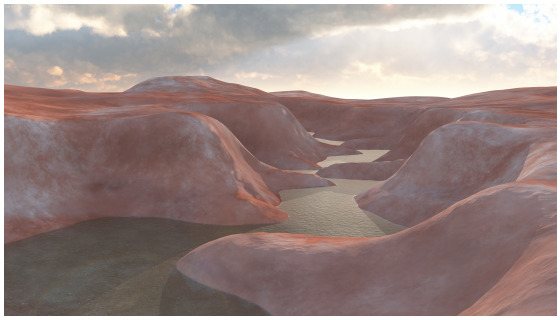

**Figure 1:** *Exemple de terrain créé à l'aide de notre modèle de représentation. Pour représenter le relief, une trentaine de primitives ont été combinées. L'élévation de l'eau est définie à l'aide d'un arbre de construction supplémentaire.*

ckés sous la forme de cartes de hauteurs qui peuvent être converties, plus tard, en des maillages adaptés à une visualisation rapide avec des mécanismes de niveaux de détails.

Dans ce travail, nous sommes partis de la constatation suivante : les paysages réels sont composés d'une structure qui rassemble plusieurs éléments géologiques et géographiques homogènes (*landform features*) qui peuvent provenir de différentes échelles. On remarquera alors qu'un terrain est la composition de rivières, de montagnes, de falaises, de lacs, de plaines. D'autres éléments, dus à l'activité humaine, peuvent structurer ces reliefs comme le terrassement dû aux routes et aux structures urbaines. Nous souhaitons donc concevoir un modèle de représentation qui repose sur la composition intuitive d'éléments caractéristiques et qui s'appuie sur des contenus générés procéduralement.

Dans cet article, nous proposons un nouveau modèle hiérarchique compact qui combine des primitives procédurales, représentant des éléments caractéristiques et géologiquement homogènes du terrain. Ceci nous permet de définir un terrain continu (Fig. 1), avec un niveau de détails variable. Notre approche permet de construire une grande variété de terrains, et de les modifier à l'aide d'un contrôle intuitif. Notre représentation repose sur une structure d'arbre qui combine les différentes primitives à l'aide d'un ensemble d'opérateurs. Les primitives et les opérateurs définissent une fonction d'élévation continue et Lipschitzienne et une fonction de poids qui correspond au domaine d'évaluation.

Cette représentation est compacte en mémoire, efficace quand il s'agit de l'évaluer, et facile à implémenter, y compris sur GPU. La définition mathématique des primitives et des opérateurs permet de visualiser le terrain de façon efficace. L'utilisation de fonctions d'élévation Lipschitziennes permet de visualiser nos terrains en temps-réel, avec un algorithme de *sphere tracing* ou par des méthodes de maillage GPU (Fig. 2). Ainsi, notre modèle autorise une représentation et une édition interactive de grandes scènes. Combinée à l'utilisation d'opérations d'éditions intuitives qui s'intéressent au placement et à la distribution d'éléments géologiques caractéristiques et homogènes, cette représentation permet un contrôle aussi bien global que local.

## 2. État de l'art

Dans cette partie, nous présentons les principales méthodes de génération procédurale de terrain, les algorithmes de simulation et les outils d'édition interactifs. Pour un état de l'art complet et récent de la modélisation procédurale, nous renvoyons le lecteur à [STBB14]. Pour un état de l'art sur les structures de données permettant la représentation de terrain, nous renvoyons le lecteur à [NLP*13].

**La modélisation procédurale** est utilisée depuis longtemps en informatique graphique. Ces méthodes sont souvent simples à implémenter et permettent de générer une infinité de terrains en modifiant simplement quelques paramètres. Les algorithmes de subdivisions adaptatives qui permettent d'obtenir un grand niveau de détails sont à classer avec les approches à base de combinaisons de fonctions de bruit à plusieurs échelles [EMP*98]. Bien que ces méthodes puissent générer des terrains infinis avec un niveau de précision illimité, elles sont difficilement contrôlables et produisent souvent des terrains sans structuration géographique.

Plusieurs algorithmes permettent d'ajouter des rivières à des terrains procéduraux. Kelley et al. [KMN88] proposent de créer des bassins versants par un algorithme de subdivision. Prusinkiewicz et al. [PH93] combinent l'algorithme du *midpoint displacement* avec un L-système pour générer des rivières fractales. Belhadj et Audibert [BA05] modifient un algorithme de subdivision stochastique pour qu'il soit contraint par les rivières et les crêtes générées par des particules suivant un mouvement Brownien. Theo [Teo09] génère un réseau de rivières avant de construire un terrain et Derzapf et al. [DGGK11] subdivisent des réseaux hydrographiques sur des planètes entières. Récemment, Génevaux et al. [GGG*13] s'inspirent de la géologie pour générer des îles parcourues de cours d'eau qui respectent les grands principes de l'hydrologie. Notre modèle de représentation s'inspire de la structure de données présentée dans cet article et l'étend en proposant de nouveaux opérateurs et de nouvelles primitives. De plus, notre modèle offre des outils d'édition intuitifs et permet, grâce aux propriétés mathématiques des fonctions Lipschitziennes, une visualisation interactive à l'aide d'un algorithme de *sphere tracing*.

**Les techniques de simulation physique** imitent l'effet de l'eau, du vent ou encore de la gravité pour sculpter des terrains. Le premier article à introduire une notion d'érosion hydraulique et de stabilisation est probablement [MKM89] et de nombreux articles comme [Nag98, CMF98, BF02] ont amélioré cette approche. La majorité des techniques décrites s'appuient sur des simulations Eulériennes qui reposent soit sur des cartes de hauteurs, soit sur des structures à base de piles de matières [BF01], soit sur des grilles régulières 3D [BTHB06]. Les méthodes Lagrangiennes qui utilisent des particules ont été étendues pour prendre en compte l'érosion dans [KBKS09], et la simulation de la corrosion a été ajoutée dans [WCMT07]. Ces algorithmes de simulation sont à la fois peu contrôlables, et très coûteux en temps de calcul et ne peuvent, par conséquent, pas être utilisés sur de très grands terrains même quand ils sont transposés sur GPU [MDH07, VBHS11].

**L'édition interactive** cherche à rendre intuitives la création et l'édition de terrains. Rusnell et al. [RME09] génèrent des terrains en calculant des distances par rapport à un graphe. Zhou et al. [ZSTR07] combinent des exemples de terrains réels, représentés par des cartes de hauteurs pour obtenir un nouveau terrain, mais leur méthode est limitée par les données d'entrées. Les méthodes à partir d'esquisses [GMS09, TGM12, TEC*14] et de modélisations interactives [PGMG09] permettent d'obtenir un contrôle important sur l'aspect final du terrain mais peuvent donner des résultats géologiquement incorrects. Les approches hybrides [SBBK08, VBHS11] qui essayent de combiner interactions et simulations physiques sont limitées à des scènes petites ou à l'édition de terrains déjà générés.

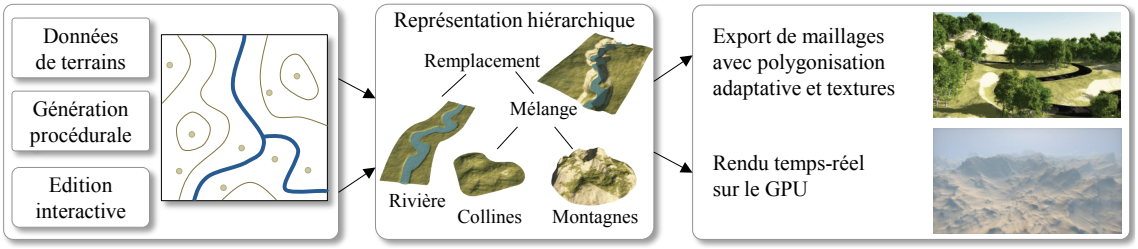


**Figure 2:** *Notre modèle de représentation permet de combiner à la fois des données réelles et des techniques procédurales. Notre structure repose sur un arbre de construction où chaque feuille correspond à une primitive. Nous visualisons nos terrains en temps-réel sur GPU, soit par un algorithme de* sphere tracing*, soit par un maillage adaptatif [DIP14].*

## 3. Représentation par arbre de construction

Notre modèle de terrain est défini par un arbre de construction (Fig. 3). Les feuilles de l'arbre agissent comme des primitives générant chacune un morceau de terrain. Généralement, cette portion de terrain représente une unité homogène comprenant un ou plusieurs éléments caractéristiques. Les nœuds internes représentent des opérateurs qui combinent et agrègent des sous-arbres c'est-à-dire des portions de terrains plus ou moins complexes composées elles-même *in fine* de plusieurs primitives. L'élévation d'un point $\mathbf{p}$ dépend donc de la combinaison hiérarchique de toutes les primitives de l'arbre.

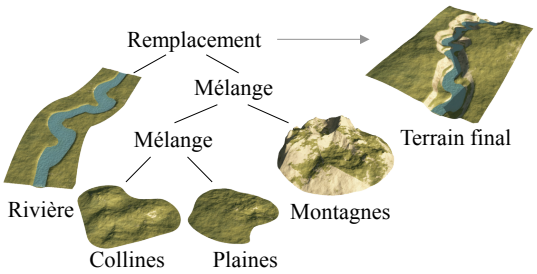


**Figure 3:** *Un arbre de construction simplifié représentant une rivière creusée dans un relief accidenté. Éditer la scène revient à modifier le placement des primitives et à manipuler les paramètres des différentes fonctions d'élévation.*

Nous définissons deux fonctions pour tout nœud : une fonction d'élévation $f : \mathbf{R}^2 \to \mathbf{R}$ et une fonction de poids $\alpha : \mathbf{R}^2 \to \mathbf{R}^+$. La fonction d'élévation $f$ est Lipschitzienne et la fonction de poids $\alpha$ est de classe $C^1$. Cette dernière définit la façon dont un morceau de terrain sera combiné avec son environnement, en utilisant un opérateur. Nous définissons le support compact, noté $\Omega_0$, de la fonction de poids $\alpha$ comme :

$$\Omega_0 = \{\, \mathbf{p} \in \mathbf{R}^2 \mid \alpha(\mathbf{p}) > 0 \,\}$$

La fonction d'élévation $f$ est définie sur $\Omega_0$. Soit $T > 0$ une valeur de seuil. Le terrain est défini sur un sous-domaine, noté $\Omega_T \subset \Omega_0$ (Fig. 4), défini par :

$$\Omega_T = \{\, \mathbf{p} \in \mathbf{R}^2 \mid \alpha(\mathbf{p}) \geq T > 0 \,\}$$

Les domaines $\Omega_0$ et $\Omega_T$ peuvent être composés par une ou plusieurs composantes connexes. Dans le reste des illustrations de cet article et pour des besoins de clarté, la région $\Omega_0$ ne sera pas montrée, sauf si le contraire est explicitement mentionné.

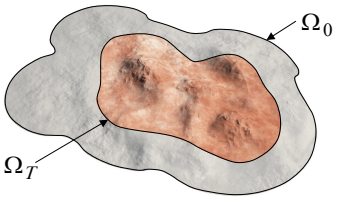


**Figure 4:** *Caractérisation d'un élément de terrain : pour tout nœud, $\Omega_0$ définit le support de définition des fonctions $f$ et $\alpha$ ; le terrain n'existe que sur $\Omega_T \subset \Omega_0$.*

Dans notre implémentation, nous utilisons $T = 1/2$. Nous définissons la surface du terrain $\mathcal{T}$ comme l'ensemble des points de l'espace dont l'élévation est définie à partir de leur projection dans $\Omega_T$ :

$$\mathcal{T} = \{\, (\mathbf{p}, f(\mathbf{p})) \in \mathbf{R}^3, \mathbf{p} \in \Omega_T \,\}$$

La construction de notre modèle hiérarchique garantit que les fonctions d'élévation $f$ des primitives et des opérateurs soient Lipchitziennes. Cette propriété nous permet d'accélérer différents calculs pour la visualisation.

**L'évaluation des normales d'un terrain** peut être utile pour certains opérateurs et est nécessaire pour le rendu. La normale à un terrain en un point $\mathbf{p}$ est calculée soit par l'évaluation du gradient de l'élévation $\nabla f$, soit par une approximation discrète.

$$\nabla f(\mathbf{p}) \simeq \frac{1}{2\varepsilon} \begin{pmatrix} f(\mathbf{p}+\varepsilon_x) - f(\mathbf{p}-\varepsilon_x) \\ f(\mathbf{p}+\varepsilon_y) - f(\mathbf{p}-\varepsilon_y) \end{pmatrix}$$

Notre système implémente un calcul exact du gradient pour les types de nœuds où cela est possible. De plus, lorsque les évaluations de $f(\mathbf{p})$, $\alpha(\mathbf{p})$ et du gradient $\nabla f(\mathbf{p})$ sont nécessaires, nous utilisons une requête optimisée qui calcule et renvoie simultanément toutes ces valeurs en un appel. Cette optimisation améliore les temps de calculs, non seulement parce que l'arbre est parcouru une seule fois, mais également parce que de nombreux termes intermédiaires sont factorisés et donc calculés une seule fois.

## 4. Primitives

Nous proposons deux types de primitives : des primitives procédurales à squelettes et des primitives construites à partir d'images. Ces deux approches sont complémentaires et permettent un contrôle différent quant à la gestion du placement des éléments caractéristiques. Les primitives procédurales ont l'avantage d'être légères en consommation mémoire et rapides à évaluer mais elles nécessitent de trouver une fonction mathématique qui imite bien le type de relief que l'on souhaite représenter. À l'inverse, les primitives construites à partir d'images permettent, à un coût mémoire plus important, d'intégrer à l'intérieur d'un terrain des données réelles ou des reliefs créés par un artiste.

### 4.1. Primitives à squelettes

Les primitives à squelettes sont inspirées à la fois du modèle CSG (*Constructive Solid Geometry*) et du modèle du *Blob-Tree* [WGG99]. Ces primitives sont définies par un squelette géométrique (point, segment, courbe, contour) et par un ensemble de paramètres qui décrivent les fonctions d'élévation et de poids en fonction de la distance au squelette. Nous utilisons différents types de squelettes adaptés à la représentation des éléments caractéristiques d'un terrain.

**Les primitives disques** représentent des portions de terrain. Le centre $\mathbf{c}$ et le rayon $r$ décrivent la zone d'influence de la primitive (Fig. 5). Soit la fonction de bruit $\eta(\mathbf{p}) : \mathbf{R}^2 \rightarrow \mathbf{R}$ qui contrôle la rugosité locale du terrain, $\{a_i\}$ un ensemble d'amplitudes de valeurs décroissantes, et $\{s_i\}$ un ensemble de fréquences croissantes. Nous définissons :

$$f(\mathbf{p}) = \mathbf{c}_z + \sum_{i=0}^{n-1} a_i \, \eta\Big((\mathbf{p}-\mathbf{c})\, s_i\Big)$$


**Figure 5:** *Les primitives disques définissent le relief dans des zones circulaires. Plusieurs fonctions d'élévation peuvent être utilisées : une fonction constante donnera une plaine, une turbulence produira des collines et une fonction de bruit* ridge-multifractals *[MKM89] imitera des montagnes.*

**Les primitives courbes** sont construites avec une courbe notée $\Gamma$ définie par morceaux et avec un ensemble de profils $\{c_i\}$ qui décrivent, le long de la courbe, le relief sous forme de coupes transversales (Fig. 6). En pratique, nous stockons chaque coupe comme une fonction uni-dimensionnelle par morceaux de quadriques afin d'accélérer les calculs.

La distance signée entre $\mathbf{p}$ et la courbe $\Gamma$ est notée $d(\mathbf{p})$, la projection de $\mathbf{p}$ sur la courbe est notée $\mathbf{q}$. Appelons $c$ le profil transversal passant par $\mathbf{q}$ et construit par interpolation selon les coordonnées curvilignes sur la courbe $\Gamma$. Nous définissons :

$$f(\mathbf{p}) = \mathbf{q}_z + c \circ d(\mathbf{p})$$

Les rivières sont créées à l'aide d'une courbe squelette $\Gamma$ qui définit la trajectoire du cours d'eau. Les profils $\{c_i\}$ permettent de représenter les plans de coupe de la rivière en fonction de son type (ruisseau, rivière, torrent, fleuve). Les confluences ou les rivières en tresses sont construites de la même manière, à l'aide de profils de coupes plus complexes qui permettent de représenter plusieurs bras ou en les combinant avec l'opérateur de mélange.

Les routes reposent sur le même modèle que les rivières mais avec des plans de coupe différents. Le squelette est une fonction continue $C^1$ approximant des morceaux de clothoïdes qui représentent la trajectoire de la route. Les profils sont paramétrés par la taille de la route et des accotements. Ces primitives définissent aussi l'élévation dans un proche voisinage de la route pour permettre le raccord avec d'autres morceaux de terrain.

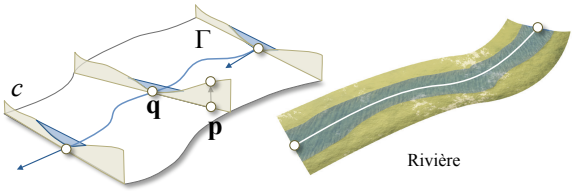


**Figure 6:** *Les primitives courbes permettent de créer facilement des reliefs suivant une trajectoire comme des routes ou des rivières. Les courbes sont construites par morceaux de cubiques et de quadriques afin d'accélérer les temps de calcul des fonctions de distance d'un point au squelette.*

**Les primitives à contours** sont créées à partir d'une courbe fermée autour d'un centre $\mathbf{c}$, d'un vecteur d'orientation $\mathbf{v}$ et d'un ensemble de fonctions de profils $\{c_i\}$ qui décrivent l'élévation de manière radiale. Chaque profil de coupe est stocké sous la forme d'une fonction uni-dimensionnelle par morceaux de quadriques (Fig. 7). Soit un point $\mathbf{p}$, nous calculons ses coordonnées polaires : la distance $d(\mathbf{p})$ et l'angle $\theta(\mathbf{p})$ calculé à l'aide du vecteur $\mathbf{v}$. L'élévation du point dépend d'une fonction de profil $c$ que l'on obtient par interpolation dans un secteur angulaire. Nous définissons :

$$f(\mathbf{p}) = \mathbf{c}_z + c \circ d(\mathbf{p})$$

Ces primitives ressemblent aux objets implicites construits par balayage de fonctions de distances anisotropes proposés par [CBS96].

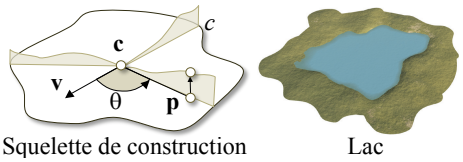


**Figure 7:** *Des primitives plus complexes peuvent être définies avec des formes polygonales. Ici, un lac est crée en définissant procéduralement des profils croissants $\{c_i\}$ de coupes transversales, étoilant un unique point* $\mathbf{c}$.

**Les fonctions de poids** sont définies comme des fonctions de classe $C^1$ décroissantes sur un support compact pour limiter l'influence de la primitive et contrôler leur influence relative dans l'arbre de construction. Soit $d(\mathbf{p})$ la distance entre le point $\mathbf{p}$ et le squelette. Nous définissons la fonction de poids $\alpha$ comme la composition de la distance avec une fonction d'atténuation $g$ :

$$\alpha(\mathbf{p}) = g \circ d(\mathbf{p})$$

Dans notre implémentation, nous utilisons la fonction définie par Wyvill dans [WGG99] qui permet un bon contrôle :

$$g(x) = \begin{cases} \left(1 - \left(\frac{x}{r}\right)^2\right)^3 & \text{si } x < r \\ 0 & \text{sinon} \end{cases}$$

### 4.2. Les primitives à base images

Les primitives à base d'images sont utilisées pour définir certains éléments caractéristiques de terrains qui seraient difficiles à modéliser par une fonction procédurale s'appuyant sur un squelette comme des morceaux de rivières détaillés ou des ridules de sables.

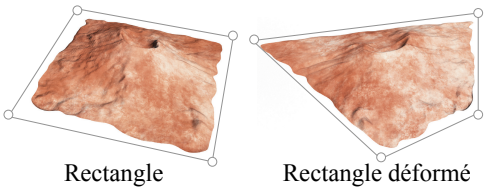


**Figure 8:** *Il est possible de projeter des données réelles sur un quadrangle convexe que l'on peut déformer. Étant donné un point dans le quadrangle, on calcule les paramètres $(u,v)$ par une interpolation bilinéaire inverse pour ensuite chercher la valeur d'élévation contenue dans l'image.*

Les fonctions d'élévation et de poids sont définies en projetant des cartes de hauteurs et de poids sur un domaine $\Omega$ dont on connaît une paramétrisation $(u,v)$. L'évaluation de $f$ et $\alpha$ se fait alors en interpolant les données pour garantir une condition de Lipschitz pour $f$ et une continuité $C^1$ pour $\alpha$. Nous proposons une paramétrisation sur des quadrangles convexes qui nous permette d'utiliser facilement des cartes de hauteurs de reliefs montagneux dans nos scènes (Fig. 8).

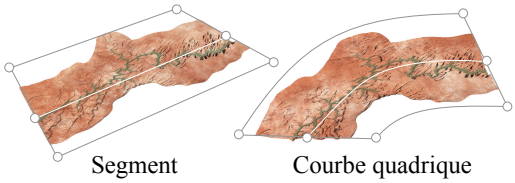


**Figure 9:** *Projeter des images de rivières sur des courbes nous permet d'utiliser une géométrie complexe plusieurs fois, sans effet apparent de répétition.*

Nous proposons également une paramétrisation le long des courbes. Ces primitives permettent en particulier de créer des rivières et des confluences aux trajectoires et aux reliefs très complexes (Fig. 9).

## 5. Opérateurs

Les opérateurs sont des nœuds internes qui combinent les fonctions d'élévation et de poids de leurs sous-arbres pour définir deux nouvelles fonctions $f$ et $\alpha$. Nous considérons par la suite des nœuds binaires et notons les deux sous-arbres A et B.

### 5.1. L'opérateur de mélange

Cet opérateur est un moyen efficace de créer de grands terrains à l'aide d'un ensemble de patchs. Le mélange de deux nœuds A et B combine les fonctions d'élévations $f_A$ et $f_B$ à l'aide des fonctions de poids $\alpha_A$ et $\alpha_B$ et permet d'agréger deux primitives de terrains (Fig. 10).

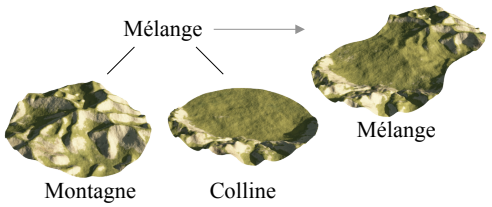


**Figure 10:** *L'opérateur de mélange permet d'agréger plusieurs primitives de manière continue. Ce nœud peut être aisément transformé en un opérateur n-aire pour gérer plusieurs primitives.*

$$f = \frac{\alpha_A f_A + \alpha_B f_B}{\alpha_A + \alpha_B} \qquad \alpha = \alpha_A + \alpha_B$$

Quand deux primitives sont suffisamment éloignées (leurs supports $\Omega_A$ et $\Omega_B$ ne s'intersectant pas), elles n'ont pas d'influence l'une sur l'autre et le résultat est alors l'union des terrains produits par les sous-arbres A et B. Quand les primitives se rapprochent, les régions d'influence s'intersectent et les primitives se mélangent progressivement (Fig. 11).

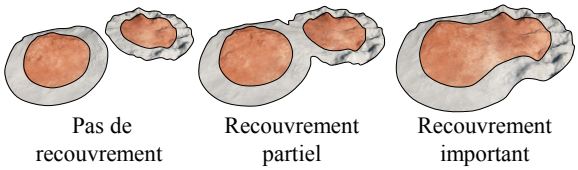


**Figure 11:** *Le mélange de deux primitives permet d'étendre le domaine de définition de notre terrain. Si les deux primitives sont disjointes, l'opérateur correspond à une union. Quand les primitives se rapprochent, leurs supports et leurs élévations se mélangent de façon continue.*

L'opérateur de mélange est commutatif et associatif et peut être facilement étendu pour devenir un opérateur $n$-aire. Cette propriété nous permet d'optimiser la taille mémoire de la structure hiérarchique mais aussi d'utiliser une grille accélératrice pour gérer de manière efficace un nombre très important de primitives.

### 5.2. L'opérateur de remplacement

Cet opérateur permet de placer un morceau de terrain caractéristique à un endroit précis. Ce nœud permet de remplacer de façon continue son sous-arbre A par le sous-arbre B.

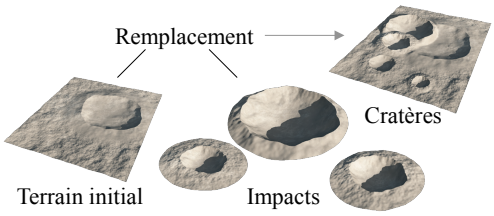


**Figure 12:** *L'opérateur de remplacement permet de facilement sculpter la forme d'un paysage. Il permet également de représenter une notion de temporalité : ici, le dernier opérande correspond au cratère le plus récent.*

Cet opérateur asymétrique peut être utilisé pour placer une géométrie précise comme un cours d'eau ou un lac sur un terrain. Il permet de facilement sculpter la trajectoire d'une route ou créer un paysage lunaire (Fig. 12). Le nœud A est remplacé par B uniquement là où $\alpha_B > 0$ :

$$f = (1-\alpha_B)\, f_A + \alpha_B\, f_B \qquad \alpha = \alpha_A$$

En général, nous voulons garder le poids de l'opérande de gauche ce qui donne $\alpha = \alpha_A$. En fonction du contexte, nous pouvons également prendre en compte l'influence de la fonction de poids $\alpha_B$ et définir alors :

$$\alpha = (1-\alpha_B)\,\alpha_A + {\alpha_B}^2$$

ce qui remplace progressivement $\alpha_A$ par $\alpha_B$.

### 5.3. L'opérateur de dépôt

Cet opérateur ajoute localement des variations et des détails sur un terrain existant A en accord avec la fonction de poids $\alpha_B$ du deuxième argument B (Fig. 13).

$$f = f_A + \alpha_B\, f_B \qquad \alpha = \alpha_A$$

De manière similaire à l'opérateur de remplacement, nous voulons souvent garder le poids de l'opérande gauche, ce qui revient à définir $\alpha = \alpha_A$. Une autre possibilité est de prendre en compte l'influence du deuxième argument $\alpha_B$ en définissant $\alpha = \alpha_A + {\alpha_B}^2$.

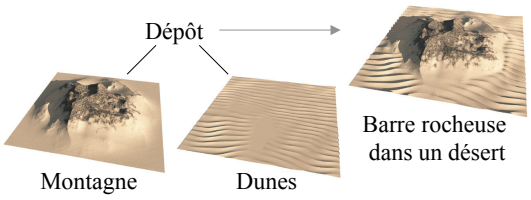


**Figure 13:** *Les dunes de sable ont été rajoutées à un paysage montagneux. Pour être présent dans les basses altitudes sans recouvrir les roches, la fonction de poids $\alpha_B$ a été définie à l'aide de l'élévation $f_A$ et du gradient $\nabla f_A$ de la montagne.*

### 5.4. L'opérateur de déformation

Cet opérateur permet une distorsion des parcelles de terrain en déformant les fonctions d'élévation $f$ et de poids $\alpha$ (Fig. 14). Une déformation est une fonction de classe $C^2$ bijective notée $\omega(\mathbf{p})$ : $\mathbf{R}^2 \to \mathbf{R}^2$. L'opérateur de déformation d'une primitive est défini comme un nœud unaire qui peut être évalué ainsi :

$$f(\mathbf{p}) = f_A \circ \omega^{-1}(\mathbf{p}) \qquad \alpha(\mathbf{p}) = \alpha_A \circ \omega^{-1}(\mathbf{p})$$

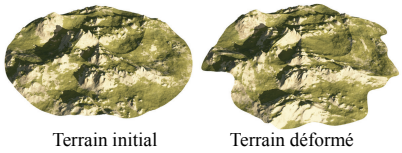


**Figure 14:** *Il est possible de déformer aussi bien les primitives que des morceaux de terrains combinés. Une déformation permet d'éliminer la répétitivité que l'on peut retrouver quand on utilise massivement un même type de primitive.*

Dans certains cas, nous voulons calculer le gradient de la fonction d'élévation, *e.g.,* pour évaluer une normale sur le terrain. Le gradient de $f$ devient :

$$\nabla f(\mathbf{p}) = \nabla f_A \circ \omega^{-1}(\mathbf{p}) \times J_{\omega^{-1}}(\mathbf{p})$$

Afin d'optimiser les temps de calcul, nous stockons le Jacobien $J_{\omega^{-1}}$ pour chaque fonction de déformation $\omega$.

Les transformations affines sont des opérateurs de déformations spécifiques. Bien que le résultat soit identique à appliquer les mêmes transformations sur les squelettes des primitives, l'avantage vient du fait qu'il est possible de transformer un morceau de terrain complexe correspondant à un sous-arbre composé de plusieurs primitives. De plus, chaque primitive peut être définie dans un repère propre et avec une orientation et une mise à l'échelle canoniques ce qui permet de les instancier plusieurs fois : notre arbre de construction devient alors un graphe orienté acyclique (Fig. 15).

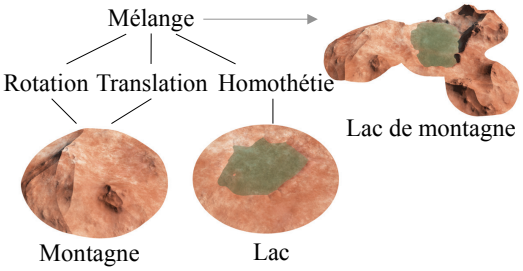


**Figure 15:** *L'instanciation associée aux opérateurs de transformations (translation, rotation, homothétie) permet de réutiliser une primitive plusieurs fois dans une scène.*

Les transformations affines peuvent être composées avec d'autres déformations. Pour des raisons d'optimisation, les transformations affines consécutives sont concaténées.

## 6. Niveaux de détails

Notre modèle de représentation à l'aide d'arbres de constructions permet d'évaluer la fonction d'élévation en intégrant un système de niveaux de détails continu. Dans notre système, nous adaptons le calcul de $f$ soit par l'utilisation d'un système de nœud interne décidant d'un niveau de détails, soit par la création de primitives paramétrées par un niveau de détails. Nous notons $\kappa(\mathbf{p}) : \mathbf{R}^2 \to [0,1]$ une fonction continue $C^1$ définissant le niveau de détails.

**Les opérateurs de niveaux de détails** sont des nœuds binaires dont les sous-arbres sont évaluées en fonction du niveau de détails voulu en entrée $\kappa(\mathbf{p})$. Ces nœuds sont caractérisés par deux valeurs faisant office de seuil : $k_A$ et $k_B$ avec $0 \leq k_A < k_B \leq 1$ qui définissent quel sous-arbre doit être évalué en fonction du niveau de détails voulu. La fonction d'élévation est définie comme suit :

$$f(\mathbf{p}) = \begin{cases} f_A & \text{si } \kappa(\mathbf{p}) \leq k_A \\ (1-t) f_A + t f_B & \text{si } k_A < \kappa(\mathbf{p}) \leq k_B \\ f_B & \text{sinon} \end{cases}$$

avec

$$t = 1 - \frac{k_B - \kappa(\mathbf{p})}{k_B - k_A}$$

La fonction de poids $\alpha$ est calculée de la même manière.

**Les primitives avec niveaux de détails continus** sont caractérisées par une fonction d'élévation $f(\mathbf{p})$ dont le calcul dépend de la fonction $\kappa(\mathbf{p})$. Ces primitives à niveaux de détails simplifient automatiquement leur évaluation pour accélérer les calculs quand le niveau de détails diminue.

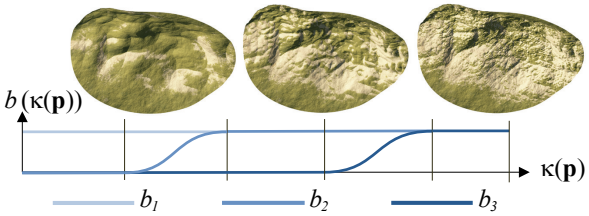


**Figure 16:** *Un exemple de primitive disque paramétrée par un niveau de détails. Alors que le niveau de détails global $\kappa$ augmente, les fonctions $b_i$ vont progressivement croître pour ajouter du détail.*

Notre système implémente différents types de primitives à niveaux de détails continus. Un exemple est la primitive disque dont la fonction d'élévation correspond à une somme de fonctions de bruits d'amplitudes décroissantes et de fréquences croissantes. Pour un faible niveau de détails, seules les plus basses fréquences sont sommées. Entre deux niveaux de détails, nous interpolons les élévations afin d'obtenir une transition continue. Nous définissons $f(\mathbf{p})$ par :

$$f(\mathbf{p}) = \mathbf{c}_z + \sum_{i=0}^{n-1} a_i \, \eta\Big((\mathbf{p} - \mathbf{c})\, s_i\Big) b_i \circ \kappa(\mathbf{p})$$

Les termes $b_i : [0,1] \to [0,1]$ sont définis comme des fonctions augmentant progressivement le niveau de détails $\kappa(\mathbf{p})$. Nous avons toujours $b_i(1) = 1$ et le niveau de détails est contrôlé en modifiant les intervalles de définition (Fig. 16) où les fonctions $b_i$ croissent progressivement de 0 à 1. Dans notre implémentation, nous définissons $\kappa(\mathbf{p})$ de manière à ce que le niveau de détails diminue alors que la distance entre la caméra et le point évalué augmente.

## 7. Visualisation

Dans cette partie, nous présentons une méthode de lancer de rayon pour visualiser notre modèle de terrain (Fig. 17) en tirant parti des propriétés de construction par arbre de fonctions. Notre approche s'inspire de la méthode de *sphere tracing* [Har94] qui est une technique robuste pour visualiser des surfaces implicites dans l'espace à l'aide d'un lancer de rayons. Contrairement aux LG-surfaces [KB89], il n'est pas nécessaire de disposer d'une fonction de classe $C^2$ : il suffit d'avoir une fonction potentiel de classe $C^0$ qui respecte la propriété de Lipschitz. La dérivée n'est pas nécessairement continue, ni même définie sur tout le domaine.

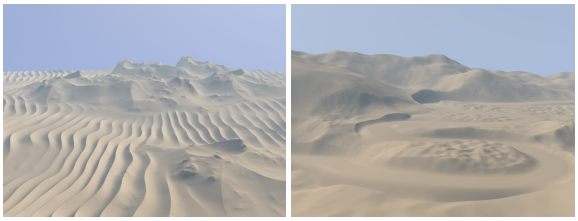

**Figure 17:** *Exemples de terrains rendus en temps-réel sur GPU avec l'algorithme de* sphere tracing. *Cette méthode permet de visualiser rapidement et sans artéfact des géométries fines et saillantes comme les crêtes des dunes de sables.*

Rappelons qu'une fonction potentiel dans l'espace $h : \mathbf{R}^3 \to \mathbf{R}$ garantit la propriété de Lipschitz sur le domaine $\Omega$ si et seulement s'il existe une constante positive $\lambda$ telle que :

$$\forall (\mathbf{p}, \mathbf{q}) \in \Omega \times \Omega, \quad |h(\mathbf{p}) - h(\mathbf{q})| < \lambda \|\mathbf{p} - \mathbf{q}\|$$

La constante de Lipschitz $\lambda$ est la valeur minimale satisfaisant cette équation. En pratique, les constantes de Lipschitz sont sur-estimées par une borne de Lipschitz.

Le *sphere tracing* consiste à marcher le long du rayon $\Delta$ avec un incrément adaptatif $|h(\mathbf{p})|/\lambda$ suffisamment petit pour ne jamais traverser la surface. Soit $r(t) = \mathbf{o} + t\,\mathbf{u}$ l'équation paramétrique du rayon ($\mathbf{u}$ normalisé). Nous notons $h(t) = h \circ r(t)$ la fonction potentiel le long du rayon et $\varepsilon$ la valeur seuil à partir de laquelle on considère qu'on intersecte la surface. L'algorithme peut s'écrire comme suit :

1. Calculer une borne de Lipschitz $\lambda$ de la fonction $h$, et l'intervalle $[t_-, t_+] = \Delta \cap \Omega$, initialiser $t$ à $t_-$.
2. Tant que $t < t_+$, calculer $h(t)$. Si $|h(t)| < \varepsilon$ alors il y a intersection, sinon progresser de l'incrément $|h(t)|/\lambda$ et continuer.

Dans les sections suivantes, nous montrons comment accélérer cet algorithme pour notre modèle de terrain.

### 7.1. Sphere tracing

Soit $\mathbf{p}$ un point de $\mathbf{R}^3$, on note $\mathbf{p}_{xy}$ et $\mathbf{p}_z$ les coordonnées $x$, $y$ et $z$ de $\mathbf{p}$ ; $f(\mathbf{p}_{xy})$ représentant l'élévation du terrain en $\mathbf{p}_{xy}$. La surface implicite représentant notre terrain correspond à la fonction potentiel notée $h : \mathbf{R}^3 \to \mathbf{R}$ et définie par :

$$h(\mathbf{p}) = \mathbf{p}_z - f(\mathbf{p}_{xy})$$

La fonction $f$ étant construite par une combinaison hiérarchique de primitives définies sur des supports compacts, les constantes de Lipschitz locales sont en général très inférieures à $\lambda$. Ces bornes peuvent être calculées en utilisant des structures de décomposition de l'espace au prix d'un surcoût mémoire élevé. Nous proposons un algorithme de *sphere tracing* adapté s'appuyant sur une évaluation à la volée d'une borne de Lipschitz locale en tout point du rayon.

L'algorithme consiste à avancer progressivement et de manière adaptative le long du rayon en calculant une borne de Lipschitz dans le voisinage du point considéré. À chaque pas, nous essayons d'avancer de la distance $\delta$. Cette distance définit un intervalle $[t, t+\delta]$ correspondant à un segment $[\mathbf{p}, \mathbf{q}]$ avec $\mathbf{q} = \mathbf{p} + \delta\mathbf{u}$ sur le rayon. Nous effectuons une requête dans l'arbre pour calculer à la fois $f(\mathbf{p})$ et la borne de Lipschitz locale $\lambda(t,\delta)$ sur l'intervalle. La fonction $h \circ r(t)$ se développe en :

$$h \circ r(t) = \mathbf{o}_z + t\,\mathbf{u}_z - f(\mathbf{a}_{xy} + t\,\mathbf{u}_{xy})$$

La dérivée de $h \circ r(t)$ par rapport à $t$ est :

$$(h \circ r)'(t) = \mathbf{u}_z - ||\mathbf{u}_{xy}||\ \ f'(\mathbf{a}_{xy} + t\,\mathbf{u}_{xy})$$

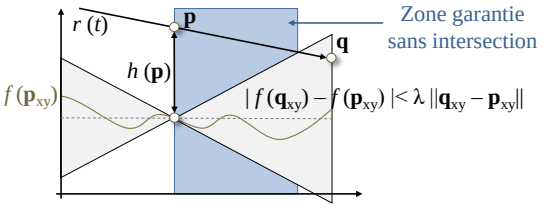


**Figure 18:** *Critère d'intersection entre un rayon et la surface du terrain.*

En fonction de la valeur de $\mathbf{u}_z - ||\mathbf{u}_{xy}||\lambda(t,\delta)$, deux cas peuvent se produire (Fig. 18 montre une interprétation géométrique des différentes pentes mentionnées dans les équations) :

1. Si $\lambda(t,\delta)\,||\mathbf{u}_{xy}|| - \mathbf{u}_z > 0$, il est possible d'avancer de la distance :
$$\frac{h(t)}{(\lambda(t,\delta)\,||\mathbf{u}_{xy}|| - \mathbf{u}_z)}$$
2. Si $\lambda(t,\delta)\,||\mathbf{u}_{xy}|| - \mathbf{u}_z \leq 0$, aucune intersection n'est détectée sur l'intervalle $[t, t+\delta]$ puisque la dérivée est positive.

Par conséquent, nous pouvons avancer le long du rayon sans intersecter la surface du terrain en progressant de la distance minimale suivante :

$$s(t,\delta) = \min\left(\delta, \frac{h(t)}{(\lambda(t,\delta)\,||\mathbf{u}_{xy}|| - \mathbf{u}_z)}\right)$$

Etant donnée une distance de progression initiale $\delta$, l'algorithme procède en trois étapes :

1. Calculer l'intervalle $[t_-, t_+] = \Delta \cap \Omega$, initialiser $t$ à $t_-$.
2. Tant que $t < t_+$, calculer $h(t)$ et $\lambda(t,\delta)$.
   2.1 Si $h(t) < 0$ alors une intersection est détectée en $t$.
   2.2 Sinon incrémenter t de la valeur de $s(t,\delta)$.
3. Mettre à jour la distance $\delta$ à $2\,\delta$.

L'étape 3 de l'algorithme contraint à essayer d'avancer constamment avec un pas de plus en plus grand. Le calcul de $s(t,\delta)$ garantit que ce pas est ajusté de manière à garantir une progression sans traverser la surface.

### 7.2. Calcul des constantes de Lipschitz

Notre modèle permet d'obtenir une évaluation des constantes de Lipschitz pour tous les types de nœuds (opérateurs) et de feuilles (primitives) de l'arbre. Par conséquent, la constante de Lipschitz $\lambda$ de la racine de l'arbre est calculée en traversant la structure de l'arbre récursivement. Pour chaque nœud nous avons besoin de calculer les élévations maximum et minimum et les constantes de Lipschitz des fonctions d'élévations et de poids.

#### 7.2.1. Primitives

Toutes les primitives ont de bonnes propriétés de continuité et de dérivabilité et respectent toujours la propriété de Lipschitz.

**Primitives à partir d'images** Pour les primitives construites à partir d'images, la fonction d'élévation est calculée par une interpolation bilinéaire d'une carte de hauteurs. Pour une projection sur un rectangle, la fonction $f$ est Lipschitzienne et sa constante est proportionelle au maximum des différences entre deux échantillons voisins de cette carte et inversement proportionelle à la distance $l$ entre les échantillons :

$$\lambda = \frac{1}{l} \sup_{i,j \in [0,n-1]^2} \left(\max\left(|f_{i,j} - f_{i+1,j}|, |f_{i,j} - f_{i,j+1}|\right)\right)$$

Lors d'une projection le long d'une courbe ou sur un quadrangle, cette constante est modifiée par la déformation.

**Primitives à squelettes** Les primitives utilisées dans notre modèle sont construites par combinaison de sommes de fonctions de bruit à différentes amplitudes et fréquences. La fonction de bruit, notée $\eta$, respecte intrinsèquement la propriété de Lipschitz, et nous notons $\lambda_\eta$ sa borne. Quand plusieurs harmoniques de bruits sont sommées comme dans le cas d'une primitive disque, nous estimons la nouvelle borne par la somme des constantes de Lipschitz de chacune des harmoniques.

$$\lambda = \sum_{i=0}^{n-1} \frac{a_i\,\lambda_\eta}{s_i}$$

**Fonctions de poids** Les fonctions de poids $\alpha$ sont définies par $\alpha(\mathbf{p}) = g \circ d(\mathbf{p})$ où $d$ est la distance euclidienne à un squelette et $g$ la fonction de Wyvill. La constante de Lipschitz de $\alpha$ est le maximum de la dérivée de $g$. Les détails sont présentés en annexe.

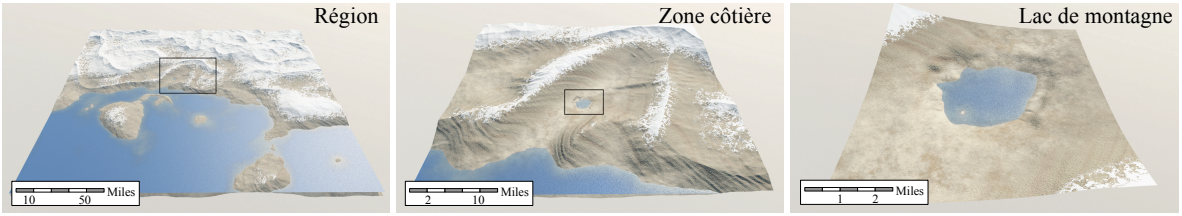


**Figure 19:** *Notre modèle permet de représenter des scènes composées d'éléments de taille très variable. Dans cet exemple, le relief général a été généré en mélangeant plusieurs primitives disques ; la montagne plus détaillée, qui borde la mer, provient d'une primitive à image. Enfin, le relief du lac a été créé par une combinaison de primitives disques et creusé par un opérateur de remplacement.*

#### 7.2.2. Opérateurs

Pour chaque type d'opérateur, nous pouvons calculer sa borne de Lipschitz en fonction des bornes des sous-arbres à combiner. Par exemple, la constante de Lipschitz $\lambda$ d'un nœud de mélange est définie comme suit :

$$\lambda \leq \max(\lambda_{f_A}, \lambda_{f_B}) + \max(\lambda_{\alpha_A}, \lambda_{\alpha_B}) \sup |f_A - f_B|$$

où $\lambda_{f_A}, \lambda_{f_B}, \lambda_{\alpha_A}, \lambda_{\alpha_B}$ représentent les constantes de Lipschitz des fonctions d'élévation et des fonctions de poids de chacun des sous-arbres respectivement. La preuve est fournie en annexe.

## 8. Résultats

Nous avons implémenté notre méthode en C++. Tous les terrains ont été générés sur un ordinateur équipé d'un Intel® Core *i*7 avec une fréquence de 3 GHz et 16 Go de RAM. Les terrains sont ensuite exportés et rendus sur MentalRay® afin de produire des images photoréalistes. Les évaluations GPU ont été réalisées avec une nVidia® GeForce GTX 670 en utilisant GLSL 4.4.

### 8.1. Édition

Notre approche permet de construire et manipuler des terrains complexes et variés en composant à la fois des primitives définissant des caractéristiques de grande taille (montagnes, plaines, plateaux) et des primitives de plus petite taille pour sculpter et éditer des détails de relief (rivières, dunes, lacs).

Un arbre de construction peut être représenté de manière compacte par un tableau de paramètres. La structure est légère et nos expérimentations montrent (Table 1) que, dans la majorité des cas, une dizaine ou une centaine de primitives par kilomètre carré suffisent pour représenter les différents types de relief présents dans la nature (collines, montagnes, rivières) ; les détails au sol étant introduits par des fonctions de bruit. Des primitives supplémentaires pourraient encore enrichir le sol de détails avec des empreintes de pas, traces de véhicules, ridules sur le sable, crevasses.

L'utilisation de primitives procédurales définies sur des domaines compacts et représentant différents types de structures géologiques/géographiques permet un contrôle à la fois local et semi-global. Il est facile pour un artiste d'éditer localement le terrain et d'augmenter les détails. Notre modèle permet de construire des scènes de tailles variées (Fig. 19) avec un niveau de détails élevé (Fig. 20).

Notre modèle peut s'étendre facilement par l'ajout de nouvelles primitives ou d'opérateurs qui répondront aux besoins spécifiques d'un utilisateur. En pratique, pour développer un nouveau nœud dans le modèle hiérarchique, il suffit d'intégrer une dizaine de lignes de code dans une nouvelle classe.

Enfin, les algorithmes de génération procédurale de terrains peuvent s'appuyer sur ce type de modèle comme le démontre la méthode de génération d'îles présentée dans [GGG*13] qui repose sur une structure similaire.

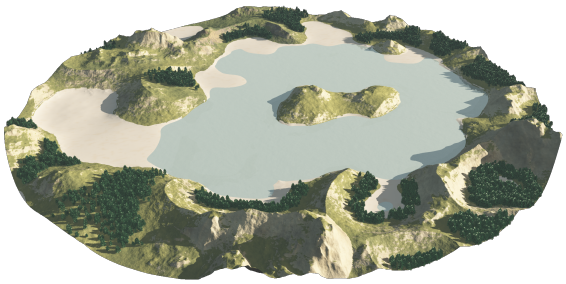

**Figure 20:** *Notre modèle permet de construire des scènes complexes. En utilisant des arbres de construction supplémentaires, il est possible de représenter plusieurs couches de matériaux (ici, la roche, le sable et l'eau). La densité de végétation peut être gérée de la même manière.*

### 8.2. Performances

Notre modèle repose sur un nombre de primitives et d'opérateurs génériques relativement réduit. Les fonctions d'élévation $f(\mathbf{p})$ et de poids $\alpha(\mathbf{p})$ sont suffisamment simples pour que leur calcul soit transposable dans un *shader program*. Les paramètres caractéristiques d'un arbre de construction et les paramètres des différents nœuds de l'arbre sont stockés dans un *buffer*. L'évaluation d'une fonction se fait en analysant le *buffer* de données et en appelant les fonctions génériques avec leurs données correspondantes.

Nous avons implémenté deux techniques différentes de

| Scène | *Sphere tracing* | | Maillage régulier | | Stockage mémoire | | | Superficie |
|---|---|---|---|---|---|---|---|---|
| | Appels à $f(\mathbf{p})$ | Temps | Appels à $f(\mathbf{p})$ | Temps | Primitives | Opérateurs | Coût | |
| Fig. 1 | 60M | 214 | 5M | 14 | 70 | 9 | 10 | 4 |
| Fig. 3 | 55M | 36 | 20M | 22 | 20 | 3 | 32 | 1 |
| Fig. 20 | 66M | 248 | 20M | 60 | 78 | 10 | 12 | 7 |
| Fig. 21 | 52M | 397 | 20M | 140 | 148 | 9 | 19 | 0.3 |
| Fig. 22 | 79M | 177 | 80M | 160 | 10 | 46 | 6 | 5 |

**Table 1:** *Statistiques pour différentes scènes. Les temps de calcul sont en millisecondes, le coût mémoire en ko, la superficie en km*$^2$*. L'algorithme de* sphere tracing *est utilisé pour générer des images de résolution FullHD (*$1920 \times 1080$*) Les scènes ont été construites en 15 à 45 minutes pour les plus complexes, la majeure partie du temps de construction étant consacrée au placement des détails (trajectoire des routes et des rivières).*

visualisation de nos terrains sur GPU : un algorithme de *sphere tracing* (Section 7) et une tesselation adaptative basée sur un *quad-tree* comme présenté dans [DIP14]. Les deux méthodes utilisent le GPU pour évaluer l'arbre de construction représentant le terrain et permettent de générer des images en un temps interactif (Fig. 17).

La tesselation adaptative basée sur un *quad-tree* permet de visualiser le terrain avec une résolution multi-échelles et sans trous apparents en utilisant un critère de niveau de détails qui dépend de la distance à la caméra. Cette technique tire avantage de la tesselation matérielle et des capacités de *culling* du GPU.

En général, l'évaluation d'un arbre de construction pour visualiser un terrain ne prend que quelques secondes sur CPU et moins d'une seconde sur GPU ce qui permet d'explorer les scènes de manière interactive. Le tableau Table 1 montre quelques statistiques sur la visualisation des différentes scènes illustrant l'ensemble de l'article. Pour les scènes composées de plusieurs centaines de primitives, l'élagage spatial de l'évaluation (grâce à une hiérarchie de boites englobantes) s'avère très utile pour diminuer le nombre de calculs à faire. L'élagage fréquentiel permet d'éviter encore plus de calculs et accélère l'algorithme de *sphere tracing* en augmentant les pas et en diminuant les constantes de Lipschitz.

## 9. Conclusions et perspectives

Nous avons proposé un modèle de représentation hiérarchique original reposant sur des primitives procédurales pour créer des terrains complexes continus avec un niveau de détails important (Fig. 21). Inspiré par les surfaces implicites à squelettes et la *Constructive Solid Geometry*, nous définissons l'élévation de chaque point en utilisant un arbre de construction dont les feuilles sont des primitives décrivant des morceaux de terrains homogènes et dont les nœuds internes représentent les opérations pour combiner et agréger ces morceaux. Le rendu des scènes est effectué sur GPU, soit par un algorithme de *sphere tracing* utilisant les propriétés mathématiques des bornes de Lipschitz que nous pouvons calculer sur l'ensemble de nos terrains, soit par une polygonisation implémentée sur GPU.

### Limitations

Notre méthode ne permet de représenter que des terrains 2D surfaciques sans surplombs, contrairement aux méthodes travaillant avec des piles de matières ou avec des voxels. Cela est dû à la nature de l'arbre et des calculs d'élévation. Toutefois, il est possible de représenter plusieurs couches de différents matériaux superposés à l'aide d'arbres supplémentaires (par exemple, l'eau et le sable dans les Fig. 20 et 22).

Afin de représenter de nouvelles variétés de paysages, notre approche nécessite d'étendre le modèle avec de nouveaux nœuds. Cette opération est facile à effectuer mais il est tout de même nécessaire d'avoir une définition analytique de chaque nouveau type de relief.

Enfin, notre modèle de représentation n'est pas adapté aux calculs de simulation tels que l'érosion. En effet, ces méthodes s'appuient sur la connaissance de voisinages qui n'est pas compatible avec notre méthode. Si cette contrainte nous fait perdre en expressivité, nous gagnons en performances en proposant une évaluation dite *pure* (qui ne requiert de calculer la fonction qu'en un seul point 2D).

### Perspectives

Les phénomènes d'érosion jouant un grand rôle dans l'aspect des terrains, nous souhaiterions tester des érosions procédurales qui ne se basent pas sur des simulations nécessitant des calculs de voisinages. Une technique d'érosion dont l'évaluation est *pure* et qui s'intégrerait dans notre modèle est proposée dans [dCB09].

Nous avons proposé un modèle de représentation qui pourrait servir de base à de nombreux travaux. Nous pensons que de nombreux algorithmes de génération procédurale existants pourraient s'appuyer sur notre modèle au lieu de travailler sur des grilles régulières. Nous souhaiterions également concevoir des outils d'édition interactive qui reposeraient sur notre modèle et permettraient à un artiste de construire une scène très rapidement.

À plus long terme, nous souhaiterions étudier comment combiner efficacement notre approche avec d'autres modèles de représentations (terrains 3D avec grottes et arches, animation de l'eau, intégration de bâtiments).

**Figure 21:** *Exemple d'une route de montagne serpentant le relief. L'artiste a esquissé la scène en mélangeant une centaine de primitives disques pour définir le relief. La trajectoire de la route a été définie à l'aide d'une vingtaine de courbes quadriques et l'opérateur de remplacement a effectué le terrassement. Afin de rendre l'intégration de la route plus naturelle, quelques primitives disques ont été placées le long de cette trajectoire pour ajuster localement l'élévation.*

### Annexe : constantes de Lipschitz

Cette partie présente les calculs de constantes de Lipschitz pour quelques opérateurs et primitives.

**Fonction de mélange de Wyvill** Nous utilisons cette fonction pour définir la fonction de poids $\alpha(\mathbf{p})$ de plusieurs primitives à squelettes. Rappelons que la fonction $C^2$ de Wyvill $g(x)$, pour un rayon de $r = 1$, est définie ainsi :

$$g(x) = \begin{cases} \left(1-x^2\right)^3 & \text{si } x < 1 \\ 0 & \text{sinon} \end{cases}$$

Pour $x < 1$, nous avons : $g'(x) = -6x(1-x^2)^2$. Trouver la constante de Lipschitz de $g$ consiste à trouver le maximum de $g'$. Pour cela, considérons $g''(x) = -6(1-x^2)(1-5x^2)$. La dérivée seconde a deux racines en 1 et $1/\sqrt{5}$. Seule cette dernière correspond à un maximum de $g'(x)$. La valeur absolue maximum de $g'(x)$ est obtenue en évaluant $g'(1/\sqrt{5})$, d'où $\lambda = 96\sqrt{5}/125$. □

**L'opérateur de mélange** Soit A et B deux sous-arbres de l'opérateur de mélange. Soit $\beta = \alpha_A/(\alpha_A+\alpha_B)$. La fonction de mélange peut alors s'écrire :

$$f = \beta f_A + (1-\beta) f_B \qquad \alpha = \alpha_A + \alpha_B$$

Le domaine est défini par $\Omega_0 = \{\mathbf{p} \in \mathbf{R}^2 \mid \alpha(\mathbf{p}) > 0\}$.

Soit $\lambda_A$ et $\lambda_B$ les constantes de Lipschitz de $f_A$ et $f_B$. Nous considérons le domaine où $f_A$, $f_B$, $\alpha_A$ et $\alpha_B$ sont de continuité $C^1$. La constante de Lipschitz $\lambda$ de $f$ est définie comme $\lambda = \sup_{\mathbf{p}\in\Omega} ||\nabla f(\mathbf{p})||$. Le gradient $\nabla f$ peut s'écrire ainsi :

$$\nabla f = \nabla\beta(f_A - f_B) + \beta\nabla f_A + (1-\beta)\nabla f_B$$

Comme $\beta \in [0,1]$, $||\nabla f|| \leq \max(\lambda_A,\lambda_B) + \lambda_\beta \sup|f_A - f_B|$. Nous devons trouver la constante de Lipschitz $\lambda_\beta$ de $\beta$. Nous avons :

$$\nabla\beta = \frac{1}{\alpha_A+\alpha_B}((1-\beta)\nabla\alpha_A - \beta\nabla\alpha_B)$$

Donc, $\lambda_\beta = \max(\lambda_{\alpha_A},\lambda_{\alpha_B})$ et finalement :

$$\lambda \leq \max(\lambda_A,\lambda_B) + \max(\lambda_{\alpha_A},\lambda_{\alpha_B})\sup|f_A - f_B|$$

Des résultats similaires peuvent être calculés sur les autres opérateurs. □

### Références


[BA05] BELHADJ F., AUDIBERT P. : Modeling landscapes with ridges and rivers : bottom up approach. In *GRAPHITE* (2005), pp. 447–450. 2

[BF01] BENEŠ B., FORSBACH R. : Layered data representation for visual simulation of terrain erosion. In *Proc. of the 17th Spring Conf. on Computer Graphics* (2001), pp. 80–85. 2

[BF02] BENEŠ B., FORSBACH R. : Visual simulation of hydraulic erosion. In *Journal of WSCG* (2002), vol. 10, pp. 79–86. 2

[BTHB06] BENEŠ B., TĚŠÍNSKÝ V., HORNYŠ J., BHATIA S. K. : Hydraulic erosion. *Comput. Animat. Virtual Worlds. Vol. 17*, Num. 2 (2006), 99–108. 2

[CBS96] CRESPIN B., BLANC C., SCHLICK C. : Implicit sweep objects. *Computer Graphics Forum. Vol. 15* (1996), 165–174. 4

[CMF98] CHIBA N., MURAOKA K., FUJITA K. : An erosion model based on velocity fields for the visual simulation of mountain scenery. *Journal of Vis. and Comp. Anim.. Vol. 9*, Num. 4 (1998), 185–194. 2

[dCB09] DE CARPENTIER G. J. P., BIDARRA R. : Interactive gpu-based procedural heightfield brushes. In *Proc. of Foundations of Digital Games* (2009), pp. 55–62. 10

[DGGK11] DERZAPF E., GANSTER B., GUTHE M., KLEIN R. : River networks for instant procedural planets. *Comput. Graph. Forum. Vol. 30*, Num. 7 (2011), 2031–2040. 2

[DIP14] DUPUY J., IEHL J.-C., POULIN. P. : *GPU Pro 5 : Advanced Rendering Techniques : Quadtrees on the GPU*. A.K Peters, 2014. 3, 10

[EMP*98] EBERT D., MUSGRAVE K., PEACHEY D., PERLIN K., WORLEY S. : *Texturing and Modeling : A Procedural Approach*. Academic Press Professional, 1998. 2

[GGG*13] GÉNEVAUX J.-D., GALIN É., GUÉRIN É., PEYTAVIE A., BENEŠ B. : Terrain generation using procedural models based on hydrology. *ACM Trans. on Graph.. Vol. 32*, Num. 4 (2013), 143 :1–143 :13. 2, 9

[GMS09] GAIN J., MARAIS P., STRASSER W. : Terrain sketching. In *Proc. of I3D* (2009), pp. 31–38. 2

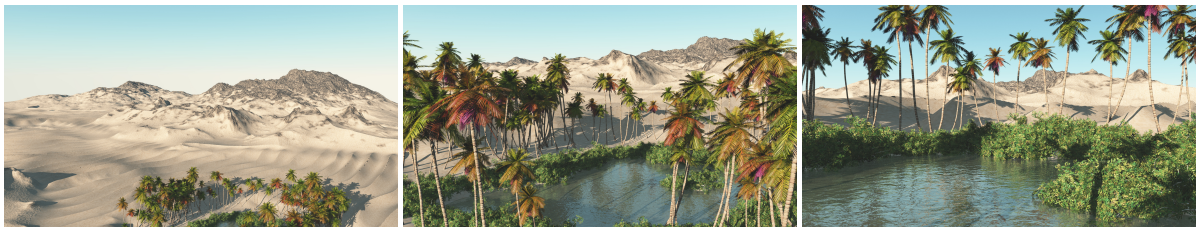

**Figure 22:** *Notre modèle de représentation, combiné à un* shading *et des textures réalistes, permet de construire des scènes très complexes. Cet arbre de construction n'utilise au total que trois types de primitives : l'oasis correspond à la Fig. 7, les dunes de sable à la Fig. 13, et les montagnes sont plusieurs instances de la Fig. 14. L'élévation d'eau et la distribution de la végétation ont été définies à l'aide d'arbres de constructions additionnels.*


[Har94] HART J. C. : Sphere tracing : A geometric method for the antialiased ray tracing of implicit surfaces. *The Visual Computer. Vol. 12* (1994), 527–545. 7

[KB89] KALRA D., BARR A. : Guaranteed ray intersections with implicit surfaces. *SIGGRAPH Computer Graphics. Vol. 23*, Num. 3 (juillet 1989), 297–306. 7

[KBKS09] KRIŠTOF P., BENEŠ B., KŘIVÁNEK J., SŤAVA O. : Hydraulic erosion using smoothed particle hydrodynamics. *Computer Graphics Forum. Vol. 28*, Num. 2 (2009), 219–228. 2

[KMN88] KELLEY A., MALIN M., NIELSON G. : Terrain simulation using a model of stream erosion. In *Proc. of SIGGRAPH* (1988), pp. 263–268. 2

[MDH07] MEI X., DECAUDIN P., HU B. : Fast hydraulic erosion simulation and visualization on GPU. In *Pacific Graphics* (2007), pp. 47–56. 2

[MKM89] MUSGRAVE F. K., KOLB C. E., MACE R. S. : The synthesis and rendering of eroded fractal terrains. In *Proc. of SIGGRAPH* (1989), pp. 41–50. 2, 4

[Nag98] NAGASHIMA K. : Computer generation of eroded valley and mountain terrains. *The Visual Computer. Vol. 13*, Num. 9-10 (1998), 456–464. 2

[NLP*13] NATALI M., LIDAL E. M., PARULEK J., VIOLA I., PATEL D. : Modeling terrains and subsurface geology. In *Proceedings of Eurographics STAR* (2013), pp. 155–173. 2

[PGMG09] PEYTAVIE A., GALIN É., MÉRILLOU S., GROSJEAN J. : Arches : a Framework for Modeling Complex Terrains. *Computer Graphics Forum. Vol. 28*, Num. 2 (2009), 457–467. 2

[PH93] PRUSINKIEWICZ P., HAMMEL M. : A fractal model of mountains with rivers. In *Graphics Interface* (1993), pp. 174–180. 2

[RME09] RUSNELL B., MOULD D., ERAMIAN M. G. : Feature-rich distance-based terrain synthesis. *The Visual Computer. Vol. 25*, Num. 5-7 (2009), 573–579. 2

[SBBK08] SŤAVA O., BENEŠ B., BRISBIN M., KŘIVÁNEK J. : Interactive terrain modeling using hydraulic erosion. In *ACM Siggraph / Eurographics Symposium on Comp. Anim.* (2008), pp. 201–210. 2

[STBB14] SMELIK R., TUTENE T., BIDARRA R., BENEŠ B. : A survey on procedural modelling for virtual worlds. *Computer Graphics Forum. Vol. 33*, Num. 1 (2014). 2

[TEC*14] TASSE F. P., EMILIEN A., CANI M.-P., HAHMANN S., BERNHARDT A. : First person sketch-based terrain editing. In *Graphics Interface* (2014), pp. 217–224. 2

[Teo09] TEOH S. T. : Riverland : An efficient procedural modeling system for creating realistic-looking terrains. In *Proc. of the ISVC : Part I* (2009), pp. 468–479. 2

[TGM12] TASSE F. P., GAIN J., MARAIS P. : Enhanced texture-based terrain synthesis on graphics hardware. *Computer Graphics Forum. Vol. 31*, Num. 6 (2012), 1959–1972. 2

[VBHS11] VANEK J., BENEŠ B., HEROUT A., SŤAVA O. : Large-scale physics-based terrain editing using adaptive tiles on the GPU. *IEEE, Computer Graphics and Applications. Vol. 31*, Num. 6 (2011), 35–44. 2

[WCMT07] WOJTAN C., CARLSON M., MUCHA P., TURK G. : Animating corrosion and erosion. In *Proc. of the Eurographics Workshop on Natural Phenomena* (2007), pp. 15–22. 2

[WGG99] WYVILL B., GUY A., GALIN É. : Extending the CSG tree - warping, blending and boolean operations in an implicit surface modeling system. *Computer Graphics Forum. Vol. 18*, Num. 2 (1999), 149–158. 4, 5

[ZSTR07] ZHOU H., SUN J., TURK G., REHG J. M. : Terrain synthesis from digital elevation models. *IEEE Transactions on Visualization and Computer Graphics. Vol. 13*, Num. 4 (2007), 834–848. 2